

# Una Herramienta para la Sincronización de Datos en Redes Móviles

**Fabrizio D. Borda Luciani; Gagliardi, Edilma Olinda**

Departamento de Informática, Facultad de Ciencias Físico Matemáticas y Naturales  
Universidad Nacional de San Luis, Argentina  
{fborda,oli}@unsl.edu.ar

**Hernández Peñalver, Gregorio**

Departamento de Matemática Aplicada, Facultad de Informática  
Universidad Politécnica de Madrid, España  
gregorio@fi.upm.es

## Abstract

Mobile networks are in apogee from more than one decade. They are composed by devices that don't always have connection with the network. On the devices can run applications that, in the event of working with other applications of the network, they have the difficulty of to maintain their up-to-date data and operate the application when the device loses the connection. To facilitate these inconveniences, a tool was developed in order to simplify the data synchronization tasks among the applications and solve inconveniences that can arise in the synchronization. This tool is divided in specific two components: one that resides in each mobile device that has the application and the other in a device of the network that centralizes the data of the applications.

**Keywords:** Data synchronize, Mobile Network, MANET Network, Mobile computation.

## Resumen

Las redes móviles están en pleno auge desde no hace más de una década. Están compuestas por dispositivos que no siempre poseen conexión con la red. Sobre los dispositivos pueden correr aplicaciones que, en caso de trabajar con otras aplicaciones de la red, tienen la dificultad de mantener sus datos actualizados y operar la aplicación cuando el dispositivo pierda la conexión. Para facilitar estos inconvenientes se desarrolló una herramienta que simplifica las tareas de la sincronización de los datos entre las aplicaciones y resolver inconvenientes que puedan surgir en la sincronización. Esta herramienta está dividida en dos componentes específicas: una que reside en cada dispositivo móvil que tiene la aplicación y la otra en un dispositivo de la red que centraliza los datos de las aplicaciones.

**Palabras claves:** Sincronización de datos, Red móvil, Red MANET, Computación móvil

## 1. INTRODUCCIÓN

Las redes móviles, o Mobile Ad-hoc NETwork (*MANET*), consisten de dispositivos inalámbricos, comúnmente conocido como *wireless*, que se comunican con los demás dispositivos sin infraestructura fija. Dos dispositivos en una red móvil pueden comunicarse, si la distancia entre

ellos es menor que la distancia mínima de alcance de los dispositivos. En este tipo de redes no es posible que dos dispositivos alejados del rango del alcance de conexión se comuniquen directamente entre ellos.

El envío de mensajes entre dos dispositivos en una MANET puede requerir un ruteo del mensaje a través de dispositivos intermedios. Esto quiere decir, que si una computadora A quiere enviarle datos a una computadora B, se debe buscar el camino, directo entre A y B o a través de otras computadoras, tal que los datos lleguen correctamente a destino [1]. Los algoritmos adecuados para el envío de información a través de una red móvil son [5] [1]: Algoritmo de ruteo Voraz, Algoritmo de ruteo por Brújula y Algoritmo de ruteo por Caras.

Debido a que estos dispositivos sobre la red móvil pueden cambiar sus ubicaciones geográficas, la conexión entre ellos no es permanente. Esto implica que, los sistemas que operan sobre estos dispositivos y necesitan enviar o solicitar información a otros sistemas de otros dispositivos, queden inoperables durante dicho lapso.

En este trabajo, se propone el desarrollo de una herramienta que permite facilitar la sincronización entre dispositivos móviles, de forma tal que los sistemas sobre los mismos puedan seguir trabajando durante el lapso que el dispositivo no posee conexión y, en el momento de retomar la conexión, el sistema cuente con sus datos actualizados.

La Organización del artículo es la siguiente: Primeramente se trata la temática de la sincronización de datos en las redes móviles. Luego, nuestra propuesta de herramienta que facilita la sincronización de datos en redes móviles. Y por último, las conclusiones y consideraciones finales.

## 2. SINCRONIZACIÓN DE DATOS EN REDES MÓVILES

Cuando un dispositivo se halla fuera del alcance de una red, la aplicación que está ejecutándose debe poder seguir operando normalmente. En ese lapso de pérdida de conexión, el usuario del dispositivo móvil y el resto de los usuarios de las demás aplicaciones de la red, siguen operando y por ende, realizan modificaciones a sus datos. En el momento que el usuario del dispositivo móvil retoma la conexión, el sistema debe actualizar tanto los datos de la aplicación local hacia el resto de las aplicaciones, como los datos de las otras aplicaciones de la red hacia esta aplicación. Esta operación recibe el nombre de sincronización de datos.

Formalmente, el termino *sincronizar* denota que un “evento ocurre al mismo tiempo que otro”. En ambientes de Computación, la sincronización de datos representa el proceso de hacer que dos conjuntos de datos sean aparentemente idénticos.

Los objetivos a cumplir para realizar la sincronización de datos en sistema de dispositivos móviles son los siguientes [8]:

- *Mantener cada sistema con los datos actualizados:* Si se piensa en una red donde sus datos están distribuidos, es indispensable que cada sistema cuente con datos actualizados.
- *Reducir la carga sobre la red:* El flujo de datos de la red puede ser reducido considerablemente si en lugar de consultar un servidor central, se accede a datos locales sincronizados.
- *Datos consistentes:* A pesar de que los clientes móviles contienen sus propios datos locales sincronizados, no siempre están conectados a la red. En esos momentos que no se posee conexión, el cliente móvil debe ser capaz de seguir operando sus sistemas con los datos locales obtenidos en la última sincronización.

- *Resolver los conflictos surgidos:* Cuando en una red existen varios clientes que utilizan los mismos sistemas sobre el mismo conjunto de datos, es posible que existan conflictos al querer sincronizar sus datos con los demás.
- *Entregar Servicio de Calidad:* Aplicaciones como voz y video, necesitan ir al mismo tiempo con cada entrega de información. Si las aplicaciones no pueden entregar el mínimo servicio de calidad requerido, las aplicaciones no podrán operar en forma óptima.

Continuando, basados en cómo será la sincronización, se puede clasificar en dos categorías [8]: Sincronización Lenta (*Slow sync*) y Sincronización Rápida (*Fast sync*).

- *Sincronización Lenta:* Durante la sincronización lenta, también conocida como *sincronización completa*, todas las entradas de un dispositivo son enviadas hacia el otro dispositivo. Una vez enviadas todas las entradas, se descartan todas las entradas duplicadas y se detectan las nuevas o modificadas para finalmente actualizar los restantes dispositivos. Este es un proceso largo y costoso en recursos, pero a veces necesario según el dispositivo.
- *Sincronización Rápida:* La sincronización rápida involucra el envío de los datos que sólo han sido modificados. Las entradas modificadas sobre el servidor pueden ser rastreadas en intervalos de tiempo. Similarmente en el cliente, se puede utilizar un indicador para saber si la entrada ha sido modificada. Mientras se realiza la sincronización, todas las entradas en el cliente son enviadas y una vez terminada la sincronización exitosa, se eliminan esos indicadores de las entradas. Este mecanismo puede provocar la existencia de conflictos durante la sincronización. Algunos conflictos pueden ser resueltos automáticamente combinando ambas entradas y otros requieren que sus conflictos sean resueltos manualmente.

La sincronización de datos sobre dispositivos móviles no es tarea sencilla. Al encontrarse sus datos dispersos sobre los dispositivos móviles y la pérdida de conexión que sufren los mismos, es probable que se produzcan colisiones y conflictos entre los datos a sincronizar. Esto último y otros temas más, los integra la herramienta que se propone en este informe la cual se detalla a continuación.

### 3. ASPECTOS GENERALES DE LA HERRAMIENTA DE SINCRONIZACIÓN DE DATOS

Para tratar los problemas de sincronización de datos y los problemas ocasionados en las aplicaciones por la pérdida de conexión en los dispositivos móviles, presentamos una herramienta integral que permite darle solución a los problemas que surgen por tal pérdida de conexión con el resto de la red móvil.

Esta herramienta tiene como objetivos, entre otros, el de administrar y facilitar las tareas de la sincronización de datos entre aplicaciones móviles que conforman la red móvil.

#### 3.1 Arquitectura para Aplicaciones Móviles

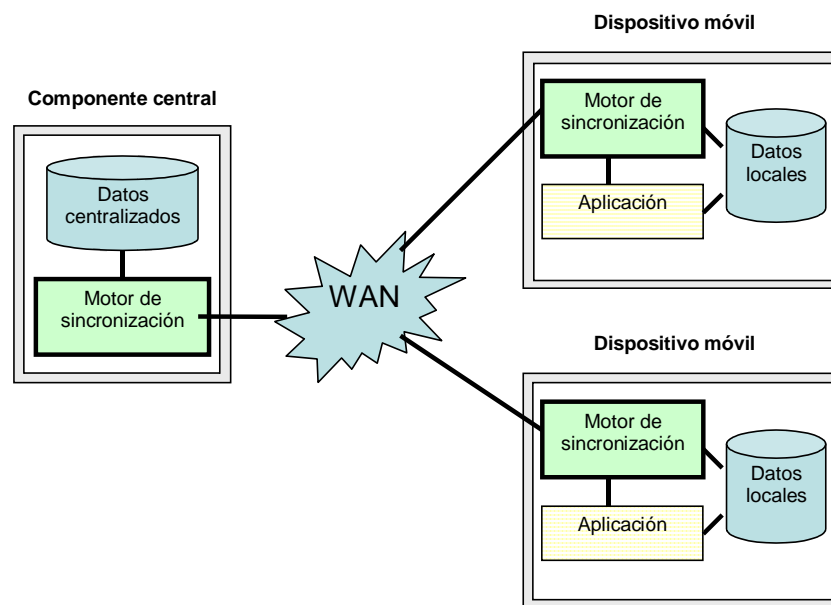
La arquitectura que proponemos para las aplicaciones móviles consiste en incorporar un administrador de la sincronización de datos, que denominamos *Motor de Sincronización*, cuya función es permitir trabajar a las aplicaciones locales sin conexión y administrar las tareas de la actualización de los datos.

Este componente, motor de sincronización, está formado a su vez de subcomponentes o módulos que pueden trabajar independientemente: uno que trabaja con la aplicación local del dispositivo móvil y el otro que centraliza los datos de las aplicaciones.

La componente del motor de sincronización que reside en el dispositivo móvil, trabaja directamente con la aplicación local del mismo y cuenta con lógica que le permite operar independientemente al estado de la conexión del dispositivo.

La otra componente del motor de sincronización, es la encargada de centralizar los datos de las aplicaciones de los dispositivos móviles, denominada *componente central*.

Como se puede apreciar en la Figura 1, tanto en la componente central, como en la componente del dispositivo móvil, se cuenta con un motor de sincronización. Con el fin de lograr diferenciarlos entre ellos, denominaremos *Motor de Sincronización Cliente* al módulo que reside en el dispositivo móvil; y *Motor de Sincronización Servidor* al módulo que está en la componente central. Tales motores de sincronización serán detallados en el punto 4.



**Figura 1: Arquitectura de aplicaciones móviles con motor de sincronización**

### 3.2 Mecanismo de Sincronización de datos

La herramienta posee un mecanismo de sincronización de datos que se basa en el envío de los datos modificados desde la última sincronización realizada exitosamente.

Para ejecutar esta sincronización, los datos almacenados en el motor de sincronización servidor tienen asignado un número de versión. Este valor se va incrementando cada vez que el dato sufre una modificación y es confirmado en la componente central.

Cuando el motor de sincronización cliente consulta a la componente central si existen nuevos datos por actualizar, el motor de sincronización cliente envía un resumen de los datos con su número de versión. De esta forma, el motor de sincronización servidor compara los números de versiones recibidos desde el dispositivo móvil con los almacenados en la componente central para determinar si los datos enviados están actualizados o necesitan actualizarse. Cuando se identifica si los datos

enviados son actuales o no, la componente central responde al motor de sincronización cliente con el estado de los datos.

### 3.3 Identificador de datos

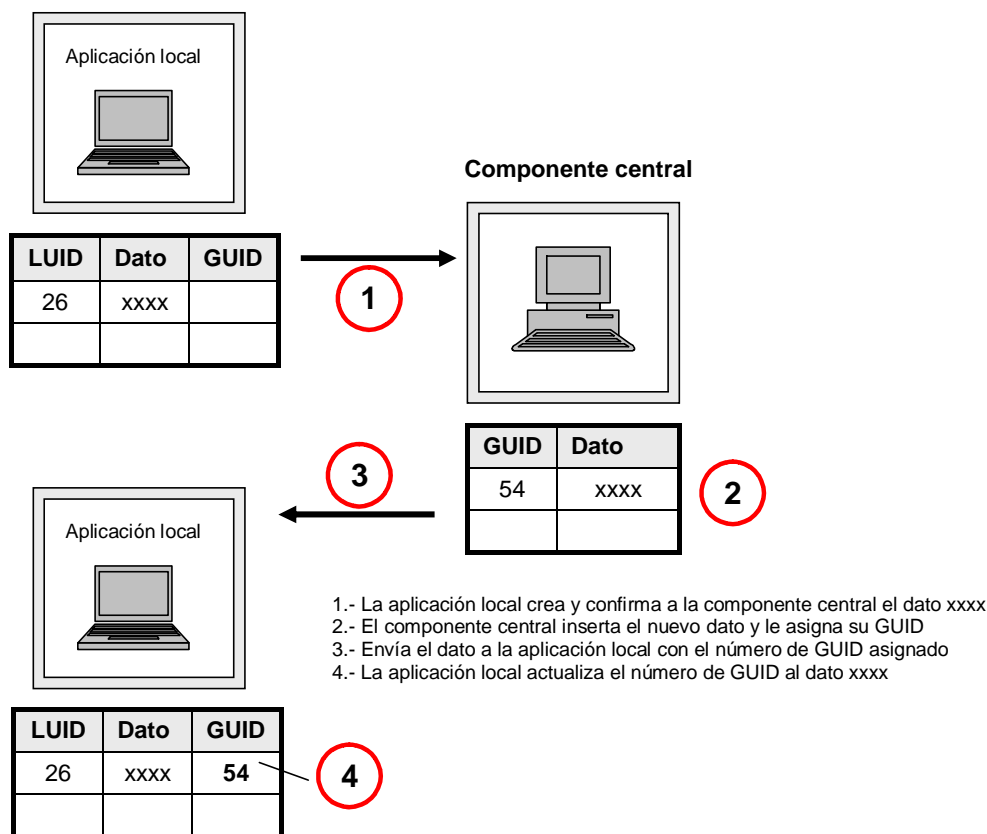
Debido a que se trabaja con datos residentes en los dispositivos móviles y datos centralizados, se requiere conocer cómo identificar los datos en sistemas móviles.

Para resolver el problema de identificar los datos en sistemas distribuidos, donde todos los sistemas pueden modificar datos localmente y luego confirmarlos a la componente central, se propone utilizar un identificador global, que denominamos *GUID*, y un identificador local, que denominamos *LUID*, que lo utiliza cada aplicación local de cada dispositivo móvil. El GUID se le otorga a cada dato existente en las aplicaciones y no puede existir que un mismo dato en dos aplicaciones distintas reciba un GUID diferente.

En la herramienta de sincronización, el encargado de asignar GUID a los datos es el motor de sincronización servidor. Este proceso se inicia cuando recibe datos nuevos desde las aplicaciones sobre los dispositivos móviles. A su vez, el motor de sincronización cliente asigna a cada dato nuevo, creado por él o producto de la sincronización de datos, el identificador LUID.

Ya que en el motor de sincronización cliente van a existir dos identificadores por cada dato distinto (el GUID y el LUID) se implementa una biyección entre los identificadores para poder recuperar un identificador en base al otro. Así, la aplicación local trabaja normalmente con el identificador LUID, hasta interactuar con el motor de sincronización servidor, donde lo hará usando el identificador GUID.

En la siguiente figura se muestra un ejemplo donde el dispositivo móvil crea un nuevo dato y como la componente central la asigna su GUID.



**Figura 2: Asignación de GUID a un dato**

### 3.4 Número de versión en datos

Supongamos que hay dos aplicaciones sobre dispositivos móviles *A* y *B*, que poseen el mismo dato. La aplicación *A* modifica este dato y lo confirma al motor de sincronización servidor. Seguidamente, también, la aplicación *A* modifica y confirma nuevamente el mismo dato. Cuando la aplicación *B* consulta si su dato está actualizado a la componente central, ésta le responde que su dato no está actualizado debido a que la aplicación *A* lo modificó en dos oportunidades, por lo que la aplicación *B* necesita actualizarlo para operar con el dato correcto y evitar que surjan conflictos.

Para llevar el control de las modificaciones que tuvieron los datos en el motor de sincronización servidor, se agrega un atributo que representa el número de versión. Este mecanismo es fundamental para las tareas de sincronización, ya que las aplicaciones locales consultan si el dato está actualizado o no, brindando su GUID y su número de versión.

En la herramienta de sincronización, el encargado de asignar e incrementar el número de versión a los datos es el motor de sincronización servidor. Por convención, cada vez que cada aplicación de un dispositivo móvil confirma la modificación de un dato a la componente central, ésta incrementa el número de versión del dato. Los datos nuevos creados por las aplicaciones locales y luego confirmados a la componente central, reciben de éste, un número de versión inicial.

En la Figura 3 se muestra un ejemplo de incremento del número de versión de un dato.

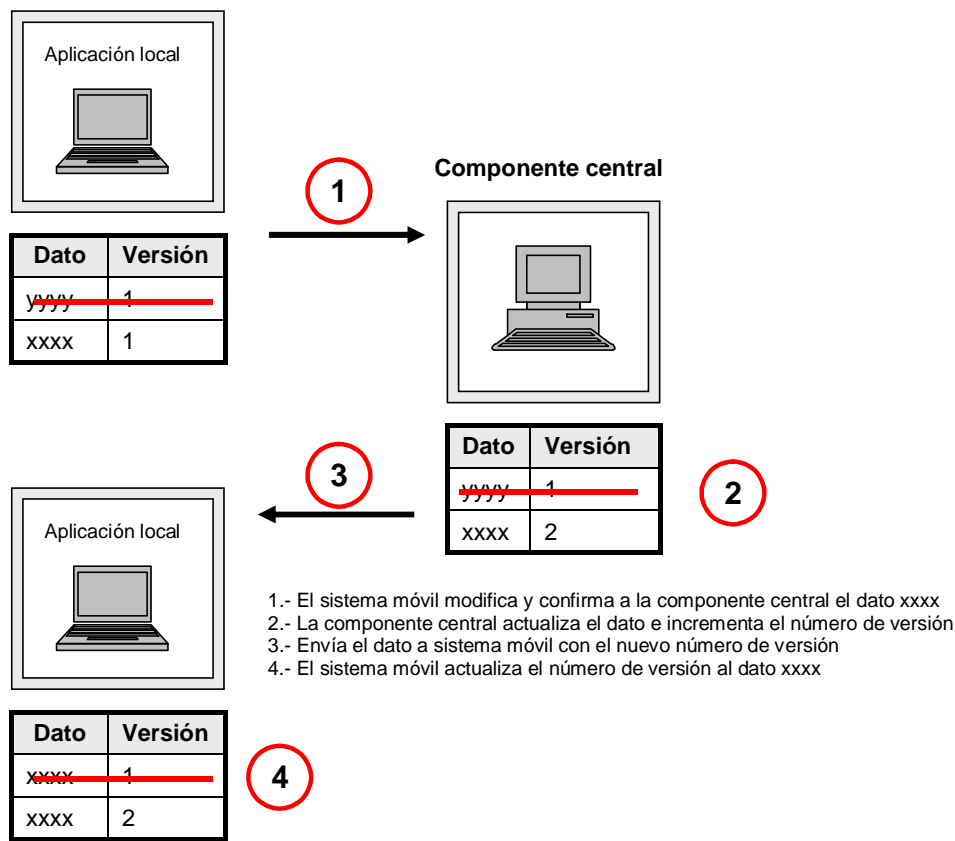


Figura 3: Incremento del número de versión de un dato

### 3.5 Manejo de conflictos

Por más precauciones que se tomen en el manejo de los identificadores de los datos y la sincronización de los mismos, hay casos donde las colisiones de datos son inevitables y los problemas surgidos son llamados *Conflicto de Datos*: una situación que debe ser resuelta durante la sincronización para evitar que los datos se corrompan.

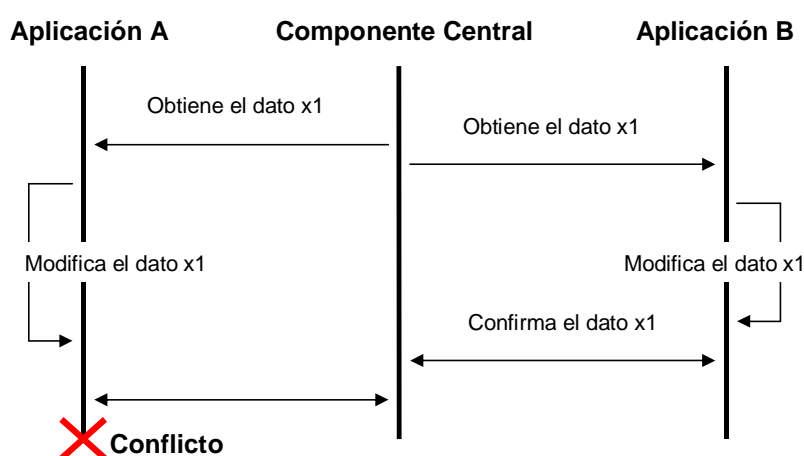
Un dato está en conflicto cuando una aplicación local de un dispositivo móvil contiene un dato modificado por ella y en la componente central existe una versión nueva del dato, que la que posee la aplicación local, posiblemente modificado y confirmado por otra aplicación.

Hay varios tipos de conflicto de datos:

- *Conflictos de Inserción*: Supongamos que se inserta un mismo dato en dos aplicaciones locales. Cuando la primera aplicación intente confirmar la inserción en la aplicación central, lo realizará sin inconvenientes. El problema lo posee la segunda aplicación al intentar sincronizar sus datos, ya que el dato insertado también fue insertado por la otra aplicación y genera problemas para determinar que dato es el válido.
- *Conflictos de Actualización*: Es similar al conflicto de inserción. Dos sistemas modifican el mismo dato en sus aplicaciones locales y al intentar sincronizar sus datos con la aplicación central se producen los conflictos.

- *Conflictos de Actualización vs. Eliminación:* Un problema frecuente en aplicaciones sobre dispositivos móviles es resolver las modificaciones de un dato que ha sido eliminada por una aplicación pero modificada por otro.

En la Figura 4 se muestra un ejemplo del surgimiento de un conflicto de actualización entre los datos de las aplicaciones. Primeramente, las aplicaciones *A* y *B* obtienen el mismo dato desde la componente central. Luego, ambas modifican ese dato y la aplicación *B* lo confirma a la componente central. Al momento de que la aplicación *A* obtiene los datos modificados en la componente central, se encuentra que el mismo dato modificado localmente fue confirmado por la aplicación *B* antes.



**Figura 4: Conflicto en los datos**

Cuando existen estos conflictos en las aplicaciones locales, el motor de sincronización cliente provee una forma de resolver estos conflictos surgidos. Para mayor seguridad y a fin de evitar datos corruptos, el motor de sincronización cliente no permite confirmar datos al servidor que estén en conflictos. Todos los conflictos se deben solucionar en la aplicación local antes de confirmar los datos a la componente central.

## 4. COMPONENTES PRINCIPALES DE LA HERRAMIENTA

Esta herramienta está dividida en dos componentes principales: el Motor de Sincronización Cliente, cuya finalidad es mantener actualizada cada aplicación de los dispositivos móviles; y el Motor de Sincronización Servidor, que tiene como objetivo centralizar todos los datos de las aplicaciones de los dispositivos móviles.

### 4.1 Componentes en el dispositivo móvil

El dispositivo móvil tiene la particularidad que la conexión con el resto de la red móvil es ocasional. Por ende, las aplicaciones sobre él, pierden la posibilidad de estar siempre conectadas con las demás aplicaciones de otros dispositivos móviles a través de la red móvil.

Cuando el dispositivo móvil no posee la conexión, el usuario debe poder operar normalmente la aplicación con sus datos locales. En el momento de recobrar dicha conexión, el sistema tiene que



poder actualizar sus datos, los cuales fueron modificados por el usuario en su aplicación, y recibir los datos actualizados de otras aplicaciones.

El motor de sincronización residente en el dispositivo móvil, o *Motor de Sincronización Cliente*, es el encargado de administrar la actualización de datos entre las aplicaciones. En un determinado momento, recibe desde la componente central, datos modificados por otras aplicaciones y envía sus datos modificados hacia la componente central.

La aplicación local siempre opera normalmente y ante cada modificación en sus datos locales, le informa al motor de sincronización dicha alteración. Con esto, el motor de sincronización cliente mantiene un registro de los datos locales que han sido cambiados para luego enviarlos a la componente central. Para ello, el motor de sincronización provee una serie de interfaces para que las aplicaciones las utilicen para informar los eventos ocurridos.

También, cuando realiza las tareas de sincronización, recibe datos actualizados desde la componente central y necesita actualizarlos para que la aplicación local disponga de ellos. Para ello, se comunica directamente con el soporte de los datos locales y las estructuras correspondientes.

Además, en caso de que los datos locales entraran en conflicto con los de la componente central, el motor de sincronización debe tener un manejador de conflicto que se encargue de tratarlos y evitar que estos conflictos provoquen información corrupta en el sistema.

El motor de sincronización cliente posee los siguientes *comandos*, las cuales pueden ser ejecutadas siempre y cuando el dispositivo móvil posea conexión con la red y pueda conectarse con la componente central.

Los comandos del motor de sincronización clientes son acciones internas que ejecuta la aplicación y son invocados por las distintas Operaciones.

Los comandos que ejecuta son:

- *Commit*: Envía datos a la aplicación local a la componente central
- *Update*: Obtiene datos desde la componente central y los actualiza en la aplicación local
- *Sync*: Actualiza los encabezados de los datos con el fin de conocer que datos de la aplicación local deber ser actualizados por otro mas nuevo y que dato local debe ser enviado a la componente central.
- *Checkout*: Obtiene desde la componente central todos los datos hacia la aplicación local.
- *Response*: Envía a la componente central el resultado de la ejecución de las operaciones.

Para su administración, el motor de sincronización cliente posee una serie de *Operaciones*. Cuando el motor de sincronización cliente realiza las tareas de sincronización, las realiza a través de las operaciones. Las operaciones se componen de varios comandos que se van ejecutando de forma secuencial.

Debido que el motor de sincronización cliente trabaja sobre dispositivo móviles, la falta de conexión puede darse de repente. Es por ello que la forma de ejecución de las operaciones es similar al comportamiento de las transacciones de una base de datos. Los cambios se confirman siempre y cuando la ejecución de la operación no haya tenido fallas. En caso de falla, los datos no sufren modificaciones. El control de la ejecución íntegra de las operaciones son realizadas en el motor de sincronización servidor.

Las operaciones que ejecuta son:

- *Checkout*: Inicializa la aplicación local con datos actualizados desde la componente central.
- *Commit*: Confirma los datos modificados por la aplicación local a la componente central
- *Detail*: Obtiene todos los detalles de un dato.
- *Synchronize*: Actualiza las diferencias entre los datos modificados localmente y los datos que posee la componente central
- *Update*: Actualiza los datos de la aplicación local por los que posee modificados la componente central
- *Conflicts*: Resuelve los conflictos de los datos locales con los de la componente central bajo alguna política.
- *Connect*: Utiliza esta operación para comprobar el estado de la conexión con la componente central.

## 4.2 Componente central

El objetivo de esta componente, es la de recolectar las modificaciones de los datos desde las aplicaciones de los dispositivos móviles. Es por ello que las aplicaciones sobre dispositivos móviles envían a la componente central los datos modificados y ésta, a su vez, le distribuye los datos a medida que se lo soliciten.

La forma de trabajar es similar a las de aplicaciones *cliente-servidor*: el cliente invoca al servidor y éste último responde al requerimiento.

El motor de sincronización de la componente central, o *Motor de Sincronización Servidor*, es el encargado de llevar el control de los datos entre las aplicaciones locales. Actúa como el núcleo central en el proceso de sincronización de datos, ya que interactúa con todas las aplicaciones locales y es el responsable del envío de la información solicitada.

El motor de sincronización servidor provee los datos solicitados por las aplicaciones locales y recibe los datos modificados por las aplicaciones de los dispositivos móviles. De esta forma, todos los datos pasan por él y centraliza la administración de los mismos.

Por ejemplo, cuando alguna aplicación local modificó o creó un nuevo dato, se comunica con la componente central para informar de dicho cambio. Así, el motor de sincronización interviene actualizando sus datos con los informados por la aplicación local y enviando la confirmación de la operación.

Otra de las características del motor de sincronización servidor es, aprovechando la centralización de los datos de las aplicaciones de los dispositivos móviles, administrar la identificación de los datos. Esto es de suma importancia, ya que es a través de estos identificadores que las aplicaciones sobre los dispositivos móviles dialogan con la componente central y viceversa.

A diferencia del motor de sincronización cliente que posee varios comandos, el motor de sincronización servidor posee sólo tres comandos propios. Esto se debe a que la forma y el manejo de la sincronización se administra desde el dispositivo móvil y este motor se dedica a responder dichos requerimientos.

Los comandos que ejecuta son:

- *Versions*: Incrementa el número de versión que poseen los datos almacenados en el motor de sincronización servidor.

- *Transactions*: Lleva un control de que una operación que modifica datos se ejecuta totalmente y sin errores.
- *Result*: Entrega al motor de sincronización cliente, resultados de la ejecución de una determinada operación.

Para su administración, el motor de sincronización cliente posee también una serie de *Operaciones*. Una operación del motor de sincronización cliente está formada por la ejecución de varios comandos. Estos comandos pueden estar en el motor de sincronización cliente o servidor. Las operaciones son iniciadas siempre por el motor de sincronización cliente y luego este, comienza la interacción con los comandos de la componente central.

Las operaciones a las que responde son:

- *Checkout*: Ante la recepción de un paquete de datos solicitando esta operación, el motor de sincronización servidor deberá ejecutar las tareas necesarias para entregar todos los datos actuales almacenados en la componente central.
- *Connect*: El motor de sincronización cliente solicita que la componente central responda el paquete con el estado de la conexión para determinar si posee conexión.
- *Commit*: El motor de sincronización cliente solicita que se confirmen datos modificados por éste. Cuando el motor de sincronización cliente recibe dicho paquete de datos, actualiza la información y debe entregar al dispositivo móvil el resultado de la ejecución de la operación.
- *Detail*: Responde a la solicitud con los detalles de los datos almacenados en la componente central.
- *Synchronize*: La operación Synchronize del motor de sincronización cliente es utilizada para actualizar este con los cambios en los datos que se deben modificar para tener toda la información actual.
- *Update*: El motor de sincronización cliente solicita que se actualicen datos almacenados en la componente central y que no posee la aplicación local. Al momento que recibe el paquete de datos, el motor de sincronización servidor debe buscar los datos almacenados y entregarlos al dispositivo móvil para que los actualice.

## 5. CONCLUSIONES Y CONSIDERACIONES FINALES

Sobre los dispositivos pertenecientes a una red móvil pueden correr aplicaciones que, en caso de trabajar con otras aplicaciones de la red, tienen la dificultad de mantener sus datos actualizados y operar la aplicación cuando el dispositivo pierde la conexión, por lo que es necesario administrar la sincronización de los datos. Entonces, se debe considerar de qué forma permitir que las aplicaciones que requieren conexión en este dispositivo puedan seguir operando sus sistemas sin conexión; y también ver la manera de incorporarse nuevamente en la red con una configuración adecuada para poder seguir operando en la misma normalmente.

A partir de la investigación realizada en distintos medios, se detectó que no existe aún una herramienta de libre acceso que facilite la sincronización de datos entre aplicaciones de dispositivos móviles.

En este informe se presentó la descripción de la herramienta que facilita las tareas de sincronización entre las aplicaciones sobre los dispositivos de la red móvil y cómo trata los problemas que estas

tareas implican. La herramienta está formada por dos componentes principales: uno que centraliza todos los datos de las aplicaciones sobre los dispositivos móviles denominado Motor de Sincronización Servidor; y el otro que reside en el dispositivo móvil, encargado de resolver los problemas de actualización de los datos de las aplicaciones llamado Motor de Sincronización Cliente.

Para el desarrollo de la herramienta, se utilizaron librerías de código abierto ya desarrolladas en lenguaje Java, con el fin de potenciar las prestaciones y acelerar el desarrollo. Por tal motivo y siguiendo la misma línea, la herramienta desarrollada es de código abierto y puesta a disposición para uso, futuras extensiones o mejoras que se le puedan realizar.

Este trabajo está enmarcado dentro de la Línea de Investigación Geometría Computacional y Bases de Datos Espacio-Temporales, perteneciente al Proyecto Tecnologías Avanzadas de Bases de Datos 22/F314, Departamento de Informática, Universidad Nacional de San Luis; en el Proyecto AL06\_PF\_013 Geometría Computacional, subvencionado por a Universidad Politécnica de Madrid; y en el marco de la Red Iberoamericana de Tecnologías del Software (RITOS2), subvencionado por CYTED.

## 6. REFERENCIAS

- [1] Beron Mario, Gagliardi Edilma Olinda, Hernández Peñalver Gregorio - “Un evaluador de algoritmos de ruteo” – Tesis de Maestría en Ingeniería del Software, Universidad Nacional de San Luis – 2006.
- [2] Beron Mario, Gagliardi Edilma Olinda, Hernández Peñalver Gregorio - “Estrategias de ruteo alternativas para redes móviles” – XI Congreso Argentino de Ciencias de la Computación, Concordia, Entre Ríos – Argentina – 2005.
- [3] Wikipedia en Español – Enciclopedia en español de uso y aporte libre - <http://es.wikipedia.org>
- [4] SyncML – Lenguaje estándar propuesto para la sincronización de información entre dispositivos móviles - <http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html>
- [5] Yvan Tupac Valdivia, Marley Vellasco y Marco Aurelio Pacheco - “An Adaptive Network Routing Strategy with Temporal Difference” – Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial
- [6] Chris Kleisath, Ojas Rege. “The Business Value Roadmap to Mobilized Software Solutions “ - <http://www.mobilizedsoftware.com/webbooks/>
- [7] Beron Mario, Gagliardi Olinda., Flores Sonia – Ruteo con Brújula en Redes sin Cables. CACIC 2001 – Congreso Argentino de Ciencias de la Computación – 2001.
- [8] David Buchmann – “SyncML (Synchronization Markup Language)” - Diploma Thesis in Informatics, University of Fribourg, Switzerland
- [9] HIBERNATE – Librería desarrollada en lenguaje JAVA que provee persistencia de objetos - <http://www.hibernate.org>
- [10] Apache Log4j – Librería de código abierto desarrollada en lenguaje JAVA que provee un servicio de monitoreo de la aplicación o logs - <http://logging.apache.org/log4j/docs/>
- [11] HSQLDB – Motor de base de datos relacional de código abierto desarrollado en lenguaje JAVA - <http://www.hsqldb.org/>